NASA/TM-2002-212129

Multirate Flutter Suppression System Design for the Benchmark Active Controls Technology Wing

Part II: Methodology Application Software Toolbox

Gregory S. Mason and Martin C. Berg
University of Washington, Seattle, Washington

Vivek Mukhopadhyay
Langley Research Center, Hampton, Virginia

December 2002

NASA/TM-2002-212129

Multirate Flutter Suppression System Design for the Benchmark Active Controls Technology Wing

Part II: Methodology Application Software Toolbox

Gregory S. Mason and Martin C. Berg
University of Washington, Seattle, Washington

Vivek Mukhopadhyay
Langley Research Center, Hampton, Virginia

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

December 2002

CONTENTS

page

Section One: Introduction ... 1
1.0 Overview ... 1
1.1 Software Requirements ... 1

Section Two: Background ... 3
2.0 Overview ... 3
2.1 A Multirate Feedback System ... 3
2.1.1 The Sampling Schedule ... 4
2.1.2 Digital Processor Gains ... 4
2.2 The Equivalent Time-Invariant System (ETIS) ... 4
2.3 Optimizing the Digital Processor Gains ... 6

Section Three: M-File Documentation ... 7
3.0 Standard Variable Definitions ... 7
3.1 Modeling M-Files ... 9
3.1.1 mr2etis ... 9
3.1.2 mr2ptv ... 9
3.1.3 ptv2etis ... 10
3.1.4 sr2etis ... 10
3.1.5 stack ... 10
3.1.6 unstack ... 11
3.2 Modeling M-Files ... 12
3.2.1 mrsim ... 12
3.2.2 mrfeedback ... 13
3.3 Synthesis M-Files ... 14
3.3.1 Input Variables ... 14
3.3.2 mropt_init ... 17
3.3.3 mropt_optim ... 18
3.3.4 mropt_extract ... 19
3.3.5 calc_LQGcost ... 19

Section Four: Synthesis Example ... 21
4.0 Problem Description ... 21
4.1 Compensator Design ... 22
4.2 Design Analysis ... 26

References ... 27
Appendix A: Global Workspace Variables ... 29
Appendix B: Support M-Files ... 31
Appendix C: M-File mkm ... 37
Appendix D: Script mropt_mkm ... 39

SECTION ONE

INTRODUCTION

1.0 OVERVIEW

The design and analysis of even a simple multirate compensator can be a complex task. In this document we describe some Matlab M-Files which aid in multirate design. They include M-Files for modeling multirate systems, for computing optimum values of a multirate compensator's gains, and for generating time domain simulations.

The remainder of the document is divided into three sections. In Section Two we review the basics of multirate modeling and our optimization algorithm. We also present the key concepts and notation which are utilized in Section Three. Section Three describes each M-File, detailing its inputs and outputs. The M-Files in this section are divided into three categories: modeling, simulation, and synthesis. Finally, in Section Four we conclude with a multirate design example.

1.1 SOFTWARE REQUIREMENTS

In addition to the standard Matlab toolbox routines, this software uses M-Files from both the Control_Toolbox and the Optimization_Toolbox. These two toolboxes must be present for the multirate software to operate.

Blank

Section Two

Background

2.0 OVERVIEW

In the following paragraphs we present the notation used in the remaining sections and review some key concepts which will be helpful to those using this software. A detailed explanation of the theory behind our M-Files can be found in References 1-5.

2.1 A MULTIRATE FEEDBACK SYSTEM

A multirate sampled-data system consists of a continuous plant in feedback with a multirate compensator. A block diagram of such a system is shown in Figure 2.1. The vector y_S in this figure represents the continuous plant sensor output. Each element of y_S can be sampled at an independent rate. The vector $\bar{y}$ represents the sampled value of y_S available to the digital processor. (In our double index notation, the discrete signal $p(m,n)$ results from sampling the continuous signal $p(t)$ at the times $t = (mN + n)T$; where the integer N is the period of repetition; T is the sampling time; m =0, 1, ... ; and n=0, 1, ..., N-1.) The digital processor obtains the current value of $\bar{y}$ and combines it with the current processor state vector, $\bar{z}$, using the state space structure shown in Figure 2.1. Each element of the processor state vector, $\bar{z}$, can be updated at an independent rate. The continuous output from the compensator, represented by the vector u, is formed by holding the output from the digital processor, $\bar{u}$, with a zero-order-hold. Each element of the vector $\bar{u}$ can be held at an independent rate to form u. The vectors v and w represent the discrete sensor and continuous process noise respectively.

Conceptually, one can divide the multirate compensator into two parts, the "sampling schedule" and the digital processor gains. This is the approach used in the synthesis and analysis software. The "sampling schedule" is a description of when each compensator input is sampled and when each output and processor state is updated, while the digital processor gains determine the dynamics of the digital processor. In the following paragraphs we discuss each in detail.

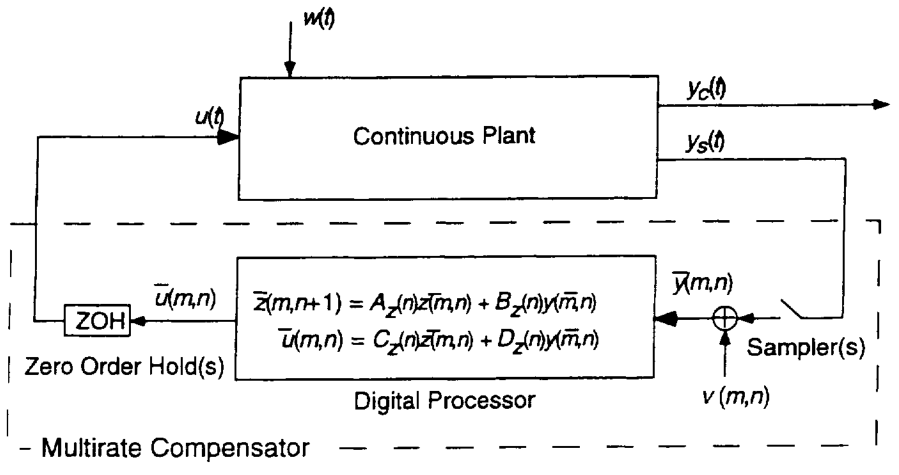


Figure 2.1 Closed-loop Multirate System

2.1.1 The Sampling Schedule

In general, the sampling and updating of y_s, $\bar{z}$, and $\bar{u}$ in Figure 2.1 can occur at any time. We, however, require that these sample and update activities occur only at integer multiples of some fixed time, called the *shortest time period* (STP). The actual value of the STP is arbitrary, but is often a function of the hardware and software used to implement the control law. We also require that the sampling and updating activities of the sensors, states and outputs repeat themselves after some fixed period of time. The period of repetition of the sampling schedule is called the *basic time period* (BTP). Finally, we define

$$\text{the integer } N = \frac{\text{BTP}}{\text{STP}} \quad \text{and the value} \quad T = \text{STP} \tag{2.1}$$

In our double index notation, the first index (m) in $p(m,n)$ indicates the integer number of BTP's which have elapsed when the sample/update occurred and the second index (n) indicates the integer number of STP's which have elapsed within the current BTP when the sample/update occurred.

We can represent the sampling schedule for the multirate compensator graphically, as shown in Figure 2.2. The figure shows a time line for each sampler, processor state, and zero-order-hold. The time line is divided into one STP increments. On the left side of the time line is a description of the signal or state, being sampled or updated. On the right side is a description of the particular activity represented by the time line, e.g. state update, sampler, or zero-order-hold. Circles on each time line indicate when a sample or update activity associated with that particular signal or state takes place. Usually the sampling schedule is shown for only one BTP since the sampling schedule repeats itself every BTP.

In most applications, the sampling/updating activities for a given sensor, output or state will be periodic *within* the BTP, as is shown in Figure 2.2. However, the sampling/updating activities do not have to be periodic within the BTP. The only requirement is that the sampling/updating activities have some period of repetition (the BTP) and that they occur at integer multiples of the STP. Once the STP and BTP have been selected, the designer can arbitrarily specify sampling/updating activities at any multiple of the STP with in one BTP. An example of a multirate sampling schedule in which the sampling/updating activities are not periodic within the BTP is shown in Figure 2.3. A sampling policy like this might be used to multiplex multiple inputs through a single analog to digital converter.

2.1.2 Digital Processor Gains

The processor gains are the values of the matrices A_Z, B_Z, C_Z, and D_Z in Figure 2.1. Like the sampling schedule, they can be periodically time-varying with a period of repetition of one BTP. Generally, these matrices are free design parameters which can be adjusted by the designer to improve the performance of the multirate compensator. The synthesis software discussed later in this document can be used to calculate optimum values for these gains.

2.2 THE EQUIVALENT TIME-INVARIANT SYSTEM (ETIS)

A multirate compensator with the structure discussed in Section 2.1 can be modeled as a periodically time-varying single-rate compensator by appending appropriate hold states to the digital processor model. This new compensator has the form

$$x(m,n+1) = A(n)x(m,n) + B(n)u(m,n) \tag{2.2a}$$

$$y_S(m,n) = C(n)x(m,n) + D(n)u(m,n) \tag{2.2b}$$

$$\text{for } n = 0,1,\ldots N\text{-}1, \text{ and } m = 0,1,\ldots$$

This compensator has a sampling period of one STP and a period of repetition of one BTP (or NT). $y_S(m,n)$ represents the values of $y(t)$ sampled every STP; $u(t)$ is formed by holding $u(m,n)$ with a zero-order-hold.

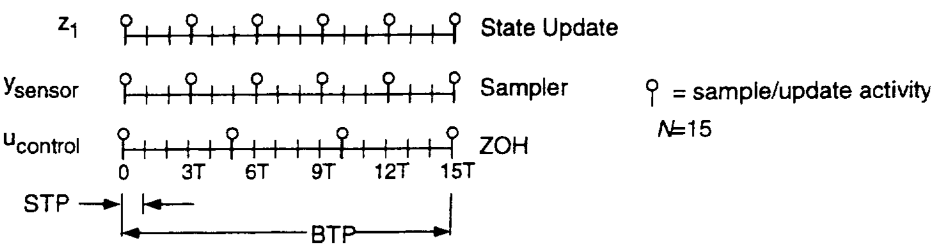


Figure 2.2. Example multirate sampling schedule with periodic sampling/updating activity

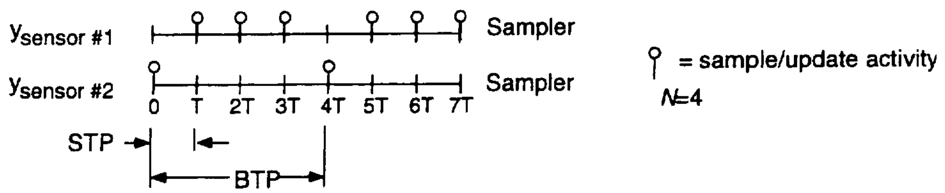


Figure 2.3. Example multirate sampling schedule with aperiodic sampling activity

The periodically time-varying single-rate compensator can be further transformed into a single-rate *Equivalent Time-Invariant System* (ETIS) with the form shown below.

$$x(m+1,0) = A_E x(m,0) + B_E u_E(m,0) \tag{2.3a}$$

$$y_E(m,0) = C_E x(m,0) + D_E u_E(m,0) \tag{2.3b}$$

where

$$\text{and} \quad y_E(m,0) = \begin{bmatrix} y_s(m,0) \\ y_s(m,1) \\ \vdots \\ y_s(m,N\text{-}1) \end{bmatrix} \quad u_E(m,0) = \begin{bmatrix} u(m,0) \\ u(m,1) \\ \vdots \\ u(m,N\text{-}1) \end{bmatrix} \tag{2.4}$$

We use the subscript E to denote vectors and matrices strictly associated with the ETIS. Notice that the ETIS input/output vectors are composite vectors containing the input/output values of the multirate (or periodically-time varying) system at N sampling times. Consequently, an ETIS is always MIMO even if the original system is SISO. If the multirate system has p inputs, q outputs and a sampling period of one STP then the ETIS is a single-rate linear *time-invariant* system with Np inputs, Nq outputs and a sampling period of one BTP.

The ETIS is fundamental to the analysis of multirate systems. It allows one to evaluate the performance and stability of complex systems comprised of multirate, periodically time-varying and/or single-rate components using only modified linear time-invariant single-rate techniques. For example, to evaluate the stability of the system in Figure 2.1, we would first transform the multirate compensator into its ETIS with a given value for N. Then we would discretize the plant at the STP of the compensator using a zero-order-hold and transform the resulting single-rate system into an ETIS using the BTP of the compensator. Next, the plant and compensator ETIS's could be combined in feedback just as if they were traditional single-rate systems. Finally, we could determine the stability of the original multirate sampled-data system from the eigenvalues of its closed-loop ETIS.

2.3 OPTIMIZING THE DIGITAL PROCESSOR GAINS

The synthesis software discussed later calculates optimum values of the digital processor gains, A_z, B_z, C_z, and D_z, by minimizing a quadratic cost function which reflects the performance of the closed-loop system in Figure 2.1. The cost function has the form

$$J = \lim_{t\to\infty} E\left\{\begin{bmatrix} y_c^T(t) & u^T(t)\end{bmatrix}\begin{bmatrix} Q_1 & M \\ M^T & Q_2\end{bmatrix}\begin{bmatrix} y_c(t) \\ u(t)\end{bmatrix}\right\} \tag{2.5}$$

where $\bar{J}$ is the cost associated with the closed-loop system shown in Figure 2.1. The vector y_c is the continuous criterion output and u is the continuous control input Q_1, Q_2 and M are the cost function weighting matrices which are selected by the designer.

This cost function has the same form as that minimized by a continuous time LQR design. Thus the cost associated with the optimized multirate compensator and that of an LQR design can be compared directly. The designer can also use this fact to help select appropriate values for Q_1, Q_2 and M.

To improve the robustness of the compensator, the optimization algorithm can optimize the digital processor gains for several different plant perturbations simultaneously. The resulting compensator will stabilize the plant at each perturbation and provide overall optimum performance. This is accomplished by minimizing the new cost function of Eqn. (2.6) which is the sum of the costs associated with each plant perturbation.

$$J = \sum_{i=1}^{\mathbf{Np}} \bar{J}_i = \sum_{i=1}^{\mathbf{Np}} \lim_{t\to\infty} E\left\{\begin{bmatrix} y_{ci}^T(t) & u_i^T(t)\end{bmatrix}\begin{bmatrix} Q_i & M_i \\ M_i^T & R_i\end{bmatrix}\begin{bmatrix} y_{ci}(t) \\ u_i(t)\end{bmatrix}\right\} \tag{2.6}$$

Here $\bar{J}_i$ is the cost associated with the i^{th} plant perturbation and there are **Np** plant perturbations.

Optimum values of A_z, B_z, C_z, and D_z, occur when

$$\frac{\partial J}{\partial A_z} = 0,\ \frac{\partial J}{\partial B_z} = 0,\ \frac{\partial J}{\partial C_z} = 0,\ \text{and}\ \frac{\partial J}{\partial D_z} = 0 \tag{2.7}$$

Our algorithm uses a gradient type numerical search to determine values of the digital processor gains such that the conditions in Eqn. (2.7) are satisfied. Because the synthesis software uses an iterative process to determine optimum values for the digital processor gains, the user must provide the software with an initial guess for A_z, B_z, C_z, and D_z. The compensator corresponding to these values must stabilize every plant perturbation considered in (2.6). Obtaining a suitable stabilizing initial guess can sometimes be a difficult problem. We refer the interested reader to Reference 1.

SECTION THREE
M-FILE DOCUMENTATION

3.0 STANDARD VARIABLE DEFINITIONS

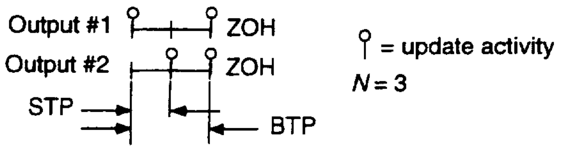
Many of the M-Files discussed in this document require similar input variables or provide similar outputs. To simplify the documentation of these M-Files a set of standard variables are used throughout this document. They are defined in Table 3.1 with Matlab variables and functions bolded.

Table 3.1 Standard Variable Definitions

Variable	Description
plt	Plant matrices in the form **plt** = $[F, G; H, J]$ where the state-space representation of the plant is $\dot{x}(t) = Fx(t) + Gu(t)$ (3.1a) $y(t) = Hx(t) + Ju(t)$ (3.1b) or $x(m,n+1) = Fx(m,n) + Gu(m,n)$ (3.1c) $y(m,n) = Hx(m,n) + Ju(m,n)$ (3.1d) depending on the value of **stp** defined later.
nplt	Number of states in (3.1), or equivalently the number of rows in F
cmp	Multirate compensator gain matrices. Given the state-space representation of the digital processor $\bar{z}(m,n+1) = A_z(n)\bar{z}(m,n) + B_z(n)\bar{y}(m,n)$ (3.2a) $\bar{u}(m,n) = C_z(n)\bar{z}(m,n) + D_z(n)\bar{y}(m,n)$ (3.2b) where m=0,1,... and n=0,1,...,N-1, so that the gain matrices, A_z, B_z, C_z and D_z, are periodically time-varying, **cmp** has the form $\mathbf{cmp} = \left[\begin{bmatrix} A_z(0) & B_z(0) \\ C_z(0) & D_z(0) \end{bmatrix}, \begin{bmatrix} A_z(1) & B_z(1) \\ C_z(1) & D_z(1) \end{bmatrix}, \ldots, \begin{bmatrix} A_z(N-1) & B_z(N-1) \\ C_z(N-1) & D_z(N-1) \end{bmatrix}\right]$ (3.3) If A_z, B_z, C_z, and D_z are constant, then $\mathbf{cmp} = \begin{bmatrix} A_z(0) & B_z(0) \\ C_z(0) & D_z(0) \end{bmatrix}$ (3.4) and it is assumed that $A_z(0)=A_z(1)= \ldots =A_z(N-1)$, $B_z(0)=B_z(1)= \ldots =B_z(N-1)$, etc. The software automatically deduces from the size of **cmp**, the value of **ncmp** and the number of compensator inputs whether the digital processor gains are periodically time-varying or not.

Continued on following page...

Table 3.1 Standard Variable Definitions *(continued from previous page)*

Variable	Description
ncmp	Number of digital processor states in Eqn. (3.2), or equivalently the number of rows in A_Z.
stp	If **stp** > 0 then **stp** is STP, the shortest sample/update period of the compensator. See Section 2.1.1, and **plt** describes a continuous plant. If **stp** = **0** then the plant matrices **plt** describe a discrete system whose sampling rate is one STP of the compensator, and **plt** describes a discrete plant.
stppbtp	Number of STP's per BTP. **stppbtp** = N. See Section 2.1.1.
su	Sampling schedule for the compensator output. **su** has two forms Case I: **su** is a matrix. **su** must have as many columns as there are compensator outputs, and must have N rows (N = BTP/STP). Each element of **su** has a value of 1 or 0. A 1 in the i^{th} column and j^{th} row of **su** indicates that the i^{th} compensator output is updated at the j^{th} STP within the current BTP. A 0 indicates no update and the previous value is held. For example, given a two output system with the following sampling schedule  Output #1 ZOH Output #2 ZOH STP BTP = update activity N = 3 the corresponding value of **su** is $\mathbf{su} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}$ (3.5) Case II: **su** is a row vector. The i^{th} column of **su** specifies the sampling period of the i^{th} compensator output hold in number of STP's. For example, **su**=[1 4] specifies that the first output is updated every STP and the second output is updated every fourth STP. This form assumes that the updating is synchronized on the first STP of the BTP.
sy	Sampling schedule for the compensator input. **sy** has the same form as **su** except that it specifies when the plant output (the compensator input) is sampled.
sz	Sampling schedule for the compensator states, **sz** has the same form as **su** except that it specifies when the compensator's digital processor states are updated.

3.1 MODELING M-FILES

3.1.1 mr2etis

Format: **[ae,be,ce,de]=mr2etis(cmp, ncmp, su, sy, sz, stppbtp)**

Description: Converts a multirate system into an ETIS with the form:

$$x(m+1,0)=\mathbf{ae}x(m,0)+\mathbf{be}\begin{Bmatrix} y(m,0) \\ y(m,n+1) \\ \vdots \\ y(m,N-1) \end{Bmatrix} \tag{3.6a}$$

$$\begin{Bmatrix} u(m,0) \\ u(m,n+1) \\ \vdots \\ u(m,N-1) \end{Bmatrix}=\mathbf{ce}x(m,0)+\mathbf{de}\begin{Bmatrix} y(m,0) \\ y(m,n+1) \\ \vdots \\ y(m,N-1) \end{Bmatrix} \tag{3.6b}$$

Inputs: **cmp, ncmp, su, sy, sz, stppbtp** The compensator description. See Table 3.1.

Outputs: **ae, be, ce, de** The ETIS matrices in Eqn. (3.6)

3.1.2 mr2ptv

Format: **[P, nu, ny, nz]=mr2ptv(cmp, ncmp, su, sy, sz, stppbtp)**

Description: Converts a multirate system into a periodically time-varying system of the form

$$x(m,n+1)=A(n)x(m,n)+B(n)y(m,n) \tag{3.7a}$$

$$u(m,n)=C(n)x(m,n)+D(n)y(m,n) \tag{3.7a}$$

Inputs: **cmp, ncmp, su, sy, sz, stppbtp** The compensator description. See Table 3.1.

Outputs: **P** The periodically time-varying system matrices in Eqn. (3.7), where

$$\mathbf{P}=\left[\begin{bmatrix} D(0) & C(0) \\ B(0) & A(0) \end{bmatrix},\begin{bmatrix} D(1) & C(1) \\ B(1) & A(1) \end{bmatrix},\ldots,\begin{bmatrix} D(N-1) & C(N-1) \\ B(N-1) & A(N-1) \end{bmatrix}\right] \tag{3.8}$$

nu	Number of inputs u in Eqn. (3.7)
ny	Number of outputs y in Eqn. (3.7)
nz	Number of states x in Eqn. (3.7)

3.1.3 ptv2etis

Format: [**ae**, **be**, **ce**, **de**]=**ptv2etis(P, nu, ny, nz, stppbtp**)

Description: Converts a periodically time-varying system into its ETIS of the form of Eqn. (3.6)

Inputs: **P, nu, ny, nz** The periodically time-varying system matrices. See Section 3.1.2
stppbtp See Table 3.1

Outputs: **ae, be, ce, de** The ETIS matrices. See Eqn. (3.6)

3.1.4 sr2etis

Format: [**ae**, **be**, **ce**, **de**]=**sr2etis(a, b, c, d, stppbtp**)

Description: Converts the single-rate system in Eqn. (3.9) into an ETIS with an N of **stppbtp**, and the form of Eqn. (3.6)

$$x(m,n+1) = a(n)x(m,n) + b(n)y(m,n) \tag{3.9a}$$
$$u(m,n) = c(n)x(m,n) + d(n)y(m,n) \tag{3.9b}$$

Inputs: **a, b, c, d** The single-rate systems matrices. See Eqn. (3.9)
stppbtp The desired N of the new ETIS. See Table 3.1

Outputs: **ae, be, ce, de** The ETIS matrices. See Eqn. (3.6)

3.1.5 stack

Format: [**y**]=**stack(w, stppbtp**)

Description: Converts a traditional discrete-time vector **w** into an ETIS vector **y**

Inputs: **w** A matrix whose columns contain the values of the vector $y(m,n)$ for Nk successive samples times, where k is an integer. The matrix **w** has the form

$$\mathbf{w} = [y(0,0), y(0,1), \dots, y(0,N\text{-}1)\ y(1,0), y(1,1), \dots, y(1,N\text{-}1), \dots, y(k\text{-}1,N\text{-}1)] \tag{3.10}$$

y is written using the double index notation of Section 2.1.
stppbtp See Table 3.1.

Outputs: **y** An ETIS vector of the values of y_E (the ETIS of y) with the form

$$\mathbf{y} = [y_E(0,0), y_E(1,0), \dots, y_E(k-1,0)] = \left[\begin{bmatrix} y(0,0) \\ y(0,1) \\ \vdots \\ y(0,N-1) \end{bmatrix} \begin{bmatrix} y(1,0) \\ y(1,1) \\ \vdots \\ y(1,N-1) \end{bmatrix} \cdots \begin{bmatrix} y(k-1,0) \\ y(k-1,1) \\ \vdots \\ y(k-1,N-1) \end{bmatrix}\right] \tag{3.11}$$

3.1.6 unstack

Format: [**w**]=**unstack(y, stppbtp**)

Description: Converts an ETIS vector **y** into a traditional discrete-time vector **w**.

Inputs: **y** An ETIS vector of the form of Eqn. (3.11)
stppbtp See Table 3.1

Outputs: **w** A vector of the values of $y(m,n)$ as in Eqn. (3.10)

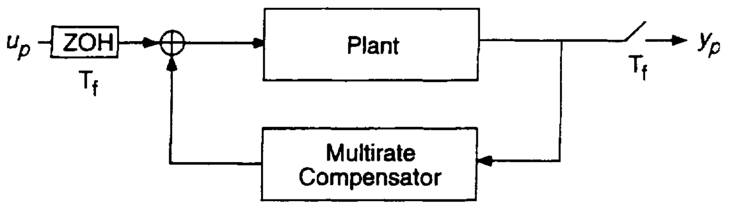
NOTE: Additional M-Files used in support of those described in this section are described in Appendix B.

3.2 SIMULATION M-FILES

3.2.1 mrsim

Format: **[y, t] =mrsim(plt, nplt, cmp, ncmp, su, sy, sz, stppbtp, simpstp, inputl, outputl, ur, stp, X0)**

Description: Generates a time response simulation of a continuous or discrete plant in feedback with a multirate compensator. The routine assumes positive feedback as shown below with the continuous inputs and outputs of interest sampled or updated every **stp/simpstp** seconds. This configuration allows the user to simulate the inter-sample response of the multirate system.



T_f = **stp/simpstp**

Inputs: **plt**, **nplt** A description of the plant, see Table 3.1.

cmp, ncmp, su, sy, sz, stppbtp,A description of the multirate compensator, see Table 3.1.

simpstp This routine can calculate the response of the closed-loop system between the sample/update activities when the plant is continuous (**stp** > 0). **simpstp** is the number of simulation steps per STP of the compensator. **simpstp** must be an integer greater than zero and **simpstp** = 1 if **stp** = 0.

inputl A vector specifying which plant inputs are connected to the compensator outputs. For example **inputl** = [1 4] indicates that the compensator outputs are connected to the first and fourth plant inputs.

outputl A vector specifying which plant outputs are connected to the compensator inputs. For example **outputl** = [2 4] indicates that the compensator inputs are connected to the second and fourth plant outputs.

ur The discrete input vector containing the values of the input at every (**stp/simpstp**) time for k·**stppbtp**·**simpstp** sample times. **ur** has the form

$$\mathbf{ur} = [u_p(0,0), u_p(0,1), \ldots, u_p(k\text{-}1, \mathbf{stppbtp}\cdot\mathbf{simpstp} - 1)] \tag{3.12}$$

Note that in Eqn. (3.12) the sample/update period is one **stp/simpstp**. This sample period should not be confused with the sample period of the multirate compensator which is simply one **stp.** The second index in the double index notation in Eqn. (3.12) takes on values between zero and **simpstp**·**stppbtp**.

stp see Table 3.1.

X0 A vector containing the initial conditions of the plant state vector. If **X0** is omitted, the initial conditions are assumed to be zero.

Outputs: **y** A vector containing the response of the plant every (**stp/simpstp**) time. **y** has the form

$$\mathbf{y} = [y_p(0,0), y_p(0,1), \ldots, y_p(k\text{-}1, \mathbf{stppbtp}\cdot\mathbf{simpstp} - 1)] \tag{3.13}$$

Note that the second index in the double index notation corresponds to the second index of Eqn. (3.12)

t a vector containing the times corresponding to the columns of **y**. When **stp** = 0 then **t** indicates the number of sampling periods.

Comments: This M-File uses the ETIS of the system to compute the time domain response. Hence, if k in Eqn. (3.12) is not a multiple of **simpstp**$\cdot N$ where (N=BTP/STP) the system response is only simulated to the nearest multiple of **simpstp**$\cdot N$

3.2.2 mrfeedback

Format: **[ae, be, ce, de] =mrfeedback(plt, nplt, cmp, ncmp, su, sy, sz,stppbtp, input1, output1, stp)**

Description: Creates a closed-loop ETIS system from a discrete or continuous plant, depending on **stp**, and a multirate compensator using *positive* feedback.

Inputs: **plt, nplt** The plant description. See Table 3.1.

cmp, ncmp, su, sy, sz, stppbtp The multirate compensator description. See Table 3.1.

input1 A vector specifying which plant inputs are connected to the compensator outputs. For example **input1** = [1 4] indicates that the compensator outputs are connected to the first and fourth plant inputs.

output1 A vector specifying which plant outputs are connected to the compensator inputs. For example **output1** = [2 4] indicates that the compensator inputs are connected to the second and fourth plant outputs.

stp See Table 3.1. If **stp** is omitted **mrfeedback** assumes **stp**=0.

Outputs: **ae, be, ce, de** The system matrices of the ETIS closed-loop system with the form of Eqn. (3.6). The inputs and outputs of this system are the inputs and outputs of the discrete plant

3.3 SYNTHESIS M-FILES

The synthesis algorithm is implemented as a series of script files. They allow the designer to optimize the performance of the closed-loop system shown in Figure 3.1 by calculating values of A_Z, B_Z, C_Z, and D_Z, such that the cost function of Eqn. (2.6) is minimized. There are three distinct steps to the optimization process: 1) entering the data which describes the problem; 2) discretizing and preprocessing the data; and 3) optimizing the compensator gain values. Each step of the optimization process is described in detail in the following paragraphs.

Although the synthesis algorithm was designed to solve the multirate sampled-data problem, it is fairly general and can be applied to a variety of related problems as well. It can compute optimum compensator gains for a sampled-data system consisting of a continuous plant in feedback with a discrete compensator as shown in Figure 2.1, or for a discrete system consisting of a discrete plant in feedback with a discrete compensator. In either case, the compensators may be single-rate or multirate and may have either time-invariant or periodically time-varying digital processor gains.

3.3.1 Input Variables

The synthesis algorithm looks for specific Matlab "workspace" variables to define the optimization problem. (A "workspace" variable is defined from the » prompt as opposed to being passed as an argument to a function.) These variables define things such as the plant dynamics, the cost function and the compensator structure and are defined in Table 3.2. The user must assign values to these variables before beginning the optimization. This can either be done manually or automatically from a script. An example of a script which defines these variables is provided in Appendix D.

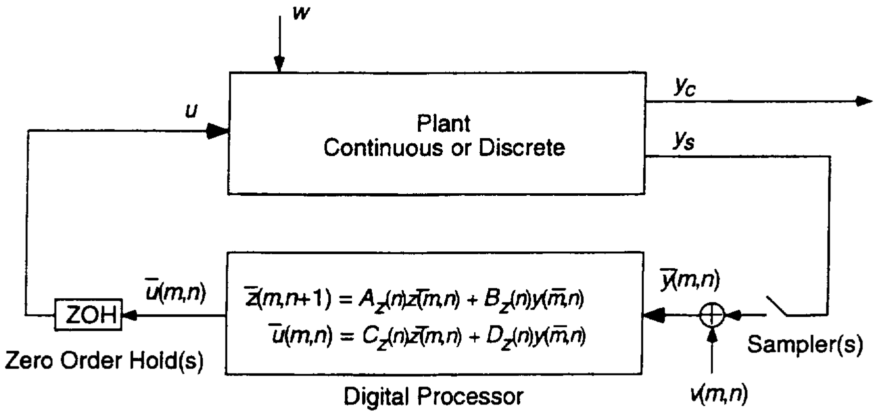


Figure 3.1 Closed-loop Multirate System

Table 3.2 Matlab Workspace Variables Defining Optimization Problem

Variable	Description
Np	The number of plant perturbations. **Np**=1 indicates only the nominal pant is defined. See Section 2.3
plt	State-space matrices of the plant(s). Given the i^{th} continuous plant perturbation $x_i(t) = F_i x_i(t) + G_i \begin{Bmatrix} u_i(t) \\ w_i(t) \end{Bmatrix}$ (3.14a) $\begin{Bmatrix} y_{ci}(t) \\ y_{si}(t) \end{Bmatrix} = H_i x_i(t) + J_i \begin{Bmatrix} u_i(t) \\ 0 \end{Bmatrix}$ (3.14b) or the i^{th} discrete plant perturbation, depending on the value of **stp** defined later, $x_i(m,n+1) = F_i x_i(m,n) + G_i u_i(m,n) + w_i(m,n)$ (3.14c) $\begin{Bmatrix} y_{ci}(m,n) \\ y_{si}(m,n) \end{Bmatrix} = H_i x_i(m,n) + J_i \begin{Bmatrix} u_i(m,n) \\ 0 \end{Bmatrix}$ (3.14d) where u is the control input; w_i is zero mean white noise input.; y_{ci} is the criterion output; and y_{si} are the outputs which are sampled by the compensator. Then $\mathbf{plt} = \left[\begin{bmatrix} F_1 & G_1 \\ H_1 & J_1 \end{bmatrix}, \begin{bmatrix} F_2 & G_2 \\ H_2 & J_2 \end{bmatrix}, \ldots, \begin{bmatrix} F_{\mathrm{Np}} & G_{\mathrm{Np}} \\ H_{\mathrm{Np}} & J_{\mathrm{Np}} \end{bmatrix} \right]$ (3.15) For convenience we assumed in Eqns. (3.14) that the input vector is partitioned into the control input u_i and the disturbance input w_i. The software, however, can accommodate a more general form for the input vector. Using the variables **ucol** and **ncol**, defined later, the user can specify which inputs correspond to control and noise inputs respectively. Any row of the input vector can be either control, disturbance or both. (We interpret an input vector specified as both control and disturbance as two separate inputs, one a control input and the other a disturbance input whose distribution vectors have the same values.) Similarly, the outputs y_{ci} and y_{si} are specified by **crow** and **srow** respectively, and any row of the output vector can be criterion output, sensor output or both. NOTE: For a discrete plant, the process noise distribution matrix is unity.
wxx	A vector of the PSD values of the white noise disturbance w_i. If $W_i \delta(\tau) = E\{w_i(t-\tau) w_i^T(t)\}$ (3.16a) or if $W_i(\delta(k) + \delta(j)) = E\{w_i(m,n) w_i^T(m+k,n+j)\}$ (3.16b) depending on the value of **stp**, then $\mathbf{wxx} = [W_1 \; W_2 \ldots W_{\mathrm{Np}}]$

continued on following page...

Table 3.2 Matlab Workspace Variables Defining Optimization Problem (*continued from previous page*)

Variable	Description
vr	The PSD values of the discrete sensor noise covariance. If the output, sampled every STP, is $y_{si}(k) = H_i x_i(k) + v_i(k) \qquad (3.17)$ where $V_i \delta(j) = E\{v_i(k) v_i^T(k+j)\}$ then $\mathbf{vr} = [V_1 \; V_2 \; ... \; V_{\mathbf{Np}}]$
ncol	A vector specifying which columns of G_i correspond to w_i. If **ncol** = [1 4], then the plant disturbance vector w is comprised of the first and fourth plant inputs.
ucol	A vector specifying which columns of G_i correspond to u_i. If **ucol** = [2 4], then the plant control input vector u is comprised of the second and fourth plant inputs. (Again, we interpret an input vector specified as both control and disturbance as two separate inputs, one a control input and the other a disturbance input whose distribution vectors have the same values.)
srow	A vector specifying which rows of C_i correspond to y_{si}. If **srow** = [1 3], then the first and third continuous plant outputs are sampled and connected to the compensator.
crow	A vector specifying which rows of C_i correspond to y_{ci}. If **srow** = [1 3], then the first and third continuous plant outputs are used to calculate the cost given in Eqn. (3.18).
q1a, **q2a**, **ma**	The cost function weighting matrices for either a continuous or discrete cost function, depending on the value of **stp** defined later. The cost function has the form $J_{continuous} = \lim_{t \to \infty} \sum_{i=1}^{\mathbf{Np}} E\left\{ \begin{bmatrix} y_{ci}^T(t) & u_i^T(t) \end{bmatrix} \begin{bmatrix} Q_{1i} & M_i \\ M_i^T & Q_{2i} \end{bmatrix} \begin{bmatrix} y_{ci}(t) \\ u_i(t) \end{bmatrix} \right\} \qquad (3.18a)$ or $J_{discrete} = \lim_{m \to \infty} \frac{1}{\mathbf{Stppbtp}} \sum_{i=1}^{\mathbf{Np}} \sum_{n=0}^{\mathbf{Stppbtp}-1} E\left\{ \begin{bmatrix} y_{ci}(m,n) \\ u_i(m,n) \end{bmatrix}^T \begin{bmatrix} Q_{1i} & M_i \\ M_i^T & Q_{2i} \end{bmatrix} \begin{bmatrix} y_{ci}(m,n) \\ u_i(m,n) \end{bmatrix} \right\} \qquad (3.18b)$ where i corresponds to a particular plant condition. $\mathbf{q1a} = [Q_{11} \; Q_{12} \cdots Q_{1\mathbf{Np}}]$ $\mathbf{q2a} = [Q_{21} \; Q_{22} \cdots Q_{2\mathbf{Np}}]$ $\mathbf{ma} = [N_1 \; N_2 \; ... \; N_{\mathbf{Np}}]$ **qa** and **ra** must be symmetric and positive semi-definite. If **ma** is left undefined its value is assumed to be zero.

continued on following page...

Table 3.2 Matlab Workspace Variables Defining Optimization Problem *(continued from previous page)*

Variable	Description
cmp	The compensator gain matrices as defined in Table 3.1. The digital processor gains can be either periodically time-varying or time-invariant. IMPORTANT: The user must provide an initial guess for **cmp** which stabilizes *all* plant perturbations.
cmpfree	This matrix specifies which of the gains in **cmp** are fixed. **cmpfree** has the same dimensions as **cmp**, but its elements are either 0 or 1. A 0 indicates that the corresponding element in **cmp** is fixed and should not be optimized. A 1 indicates that the corresponding element in **cmp** is free and can be optimized.
su, **sy**, **sz**	The sampling schedule description. See Table 3.1. By selecting appropriate values for **su**, **sy**, **sz** and **Stppbtp** the designer can specify whether the compensator is multirate or single-rate.
Stppbtp	The same as **stppbtp** in Table 3.1. Note that the first letter of **Stppbtp** is upper case, indicating this is a global workspace variable. See Section 3.3.2.
stp	STP as defined in Table 3.1. If **stp** > 0 the algorithm assumes that the plant (**plt**), the processor noise (**wxx**) and the cost weighting matrices (**q1a**, **q2a**, and **ma**) are all defined for a *continuous* plant in feedback with the discrete compensator. If **stp** $= 0$ the algorithm assumes that the plant (**plt**), the processor noise (**wxx**) and the cost weighting matrices (**q1a**, **q2a**, and **ma**) are all defined for a *discrete* plant in feedback with the discrete compensator, and that the sampling period of the discrete plant is one STP of the multirate compensator

3.3.2 mropt_init

Format: **mropt_init**

Description: This script operates on the workspace variables defined in Table 3.2. It discretizes (if necessary) the plant, process noise and cost function; and defines a set of global workspace variables used by the optimization script (**mropt_optim**). This script must be rerun after making any changes in the workspace variables defined in Table 3.2, with exception to changes in the values of **cmp** or **cmpfree**.

Inputs: **mropt_init** is a script which uses the workspace variables defined in Table 3.2.

Outputs: mropt_init save it's output to the global workspace variables defined in Appendix A.

Comment: The global workspace variables defined by **mropt_init** are used by both the optimization script **mropt_optim** and by the M-File **calc_LQGcost**. All global variables generated by **mropt_init** begin with a capital letter. This helps to differentiate them from other workspace variables. A brief description of these global variables is given in Appendix A.

3.3.3 mropt_optim

Format: **mropt_optim**

Description: **mropt_optim** computes optimum values of the digital processor gains such that the cost function defined in Eqn. (3.18) is minimized.

Inputs: **mropt_optim** uses the global workspace variables generated by **mropt_init** in addition to the three workspace variables defined in Table 3.3.

Outputs: The script returns a value of the cost function, the gradient of the cost function with respect to the free compensator parameters, and the optimum values of the digital processor gains. These values are automatically displayed on the computer screen and simultaneously stored in the workspace variables defined in Table 3.4.

Comments:
1) As already noted, if the user modifies the problem definition by changing the values of the variables defined in Table 3.2, the script **mropt_init** must be rerun before running **mropt_optim**. The exceptions are changes to the variables defined in Table 3.3.
2) **mropt_optim** can be computationally intensive. The user can abort the optimization with a Control-C key sequence. Also see **mropt_extract**, Section 3.3.4

Table 3.3 Input Workspace Variables for **mropt_optim**

Variable	Description
cmp	This variable contains the compensator gains and is the *same* variable as described in Table 3.2. These are the starting values for the optimization and must stabilize all plant perturbations.
cmpfree	This matrix specifies which of the gains in **cmp** are fixed and is the *same* variable as described in Table 3.2.
dscale	During optimization the compensator gains are scaled using a variable named **dscale** to help improve the numerical accuracy of the search. They are scaled as follows $$Xscaled = \mathbf{inv}(\mathbf{dscale})\cdot Xunscaled$$ where *Xunscaled* is an unscaled vector containing the free compensator gains and *Xscaled* are those values scaled. Typically **dscale** is a diagonal matrix chosen so that all the elements in *Xscaled* have the same magnitude. The user can either manually set **dscale** or have **mropt_optim** set it automatically. If **dscale** exists *and* has the correct dimensions then **mropt_optim** uses that value of **dscale**, otherwise a new value for **dscale**, based on the current values in **cmp**, is automatically calculated. Typing "`clear dscale`" before running **mropt_optim** forces the routine to scale the problem based on current values in **cmp**.

Table 3.4 Output Workspace Variables for **mropt_optim**

Variable	Description
cmpf	This contains the optimized values of **cmp** after **mropt_optim** has completed the optimization.
jcost	This contains the value of the cost function after **mropt_optim** has completed the optimization.
djdp	This contains a scaled value of the gradient of the cost function with respect to the compensator values after **mropt_optim** has completed the optimization.

3.3.4 mropt_extract

Format: **mropt_extract**

Description: After aborting the script **mropt_optim**, usually with a Control-C key sequence, the script **mropt_extract** can extract the value of the "optimized" compensator at the last iteration.

Inputs: None. Can only be rerun after aborting **mropt_optim**.

Outputs: The optimized values of the digital processor gains at the last iteration before **mropt_optim** was aborted are displayed on the computer and stored in the workspace variable **cmpf**.

Comments: Sometimes it is necessary to abort **mropt_optim** before is has completed the optimization. The user might wish to rescale the free parameters by changing **dscale**, or might decide to free up additional gains in **cmp** by changing **cmpfree**. The user can restart the optimization from the last iteration by setting **cmp=cmpf** and rerunning **mropt_optim**.

3.3.5 calc_LQGcost

Format: **[jcost]=calc_LQGcost(j)**

Description: Calculates the discrete LQG cost assuming a sampling period of one **stp**, This routine uses the global workspace variables computed by **mropt_init**. Use this routine to find the minimum value of the cost function attainable by **mropt_optim**

Inputs: **j** = 0 or is undefined, then the routine calculate the sum of the LQG costs for all **Np** plant conditions corresponding to the cost $J_{discrete}$ given in Eqn. (3.18).

0<j<Np then the routine calculates the LQG cost for only the **j**th plant condition

Outputs: **jcost** The value of the LQG cost function

Comments: **calc_LQGcost** uses the Controls_Toolbox routines **dlqr** and **dlqe**. They require that **ra** in Table 3.2 be positive-definite.

NOTE: Additional M-Files used in support of those described in this section are described in Appendix B.

Blank

SECTION FOUR

SYNTHESIS EXAMPLE

4.0 PROBLEM DESCRIPTION

In this section we present an example design problem which utilizes the M-Files discussed in the previous section. Our objective is to design a multirate compensator for a lightly damped mass-spring-mass system.

The mass-spring-mass (MKM) system is shown in Figure 4.1. It consists of two masses connected by a spring and damper. The control inputs are the force inputs u_1 and u_2; the sensor outputs are the displacements x_1 and x_2; and the disturbance input is w. The sensor outputs are corrupted by discrete sensor noise (not shown on figure) with covariance v. Nominal values for the plant parameters, along with those for one known plant perturbation, are given in Table 4.1. These parameter values result in a system with two poles at the origin associated with x_1, and two lightly damped poles associated with x_2. The open-loop poles are listed in Table 4.1. A M-File which defines the system matrices for the MKM system is given in Appendix C.

The goal of our design is to increase the damping coefficient on the lightly damped poles to ζ=0.707 without shifting their natural frequency, and to move the two poles at the origin so that the x_1 response has a frequency of 0.825 rad/sec with a damping coefficient ζ=0.707.

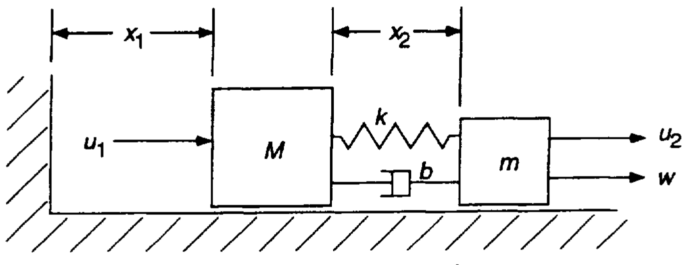


Figure 4.1 Mass-Spring-Mass System

Table 4.1 Mass-Spring-Mass Parameter Values

	Nominal Plant	Perturbed Plant
M	1.0 kg	1.0 kg
m	0.1 kg	0.2 kg
k	0.01 N/m	0.01 N/m
b	0.01 N·s/m	0.01 N·s/m
$E\{w^T(t-\tau)w(t)\}$	$0.1\delta(\tau)N^2$	$0.1\delta(\tau)N^2$
$E\{v^T(k+j)v(k)\}$	$0.001\,\delta(j)m^2$	$0.001\,\delta(j)m^2$
Open-Loop Poles	0, 0, -0.055± 3.316i	0, 0, -0.030±2.449i

4.1 COMPENSATOR DESIGN

Our compensator design followed the preceding steps. All computer inputs and outputs in the following paragraphs are in `this font`. **`Bolded text`** should be input by the user, `plain text` is returned by the computer, instructional comments are in *italic*.

Step 1: Select the weighting matrices for the quadratic cost function given in Eqn. (2.6) which characterize the desired closed-loop performance

The cost function given in Eqn. (2.6) is the sum of the costs associated with each plant perturbation. In our case we have two plants - the nominal and the perturbed plant. We determined the weighting matrices for our cost function by designing two continuous time LQ regulators which placed the closed-loop poles for the nominal and perturbed plants in the desired locations. The cost functions are

$$J_{\text{nominal}} = \mathrm{E}\{5.5x_1^2 + 2.2\dot{x}_2^2 + 10000u_1^2 + 10u_2^2\}/10.6 \tag{4.1a}$$

$$J_{\text{perturbed}} = \mathrm{E}\{6.5x_1^2 + 4.8\dot{x}_2^2 + 10000u_1^2 + 10u_2^2\}/8.0 \tag{4.1b}$$

where the costs have been scaled so that the LQR cost for each is unity. The cost function characterizing the performance of our multirate compensator is the sum of the these two cost functions. By minimizing (J_{nominal} + $J_{\text{perturbed}}$) the optimization routine will attempt to find a single compensator whose performance approaches that of the LQR designs for both the nominal and the perturbed plant. Of course we do not expect such a compensator to exist. Instead the resulting compensator will represent a compromise between good performance at the nominal plant condition and good performance at the perturbed condition.

Step 2: Select an appropriate compensator structure and sampling schedule based on the desired closed-loop performance

The closed-loop LQR system has a fast mode associated with x_1 and u_1 and a slow mode associated with x_2 and u_2. We selected a multirate compensator which capitalized on this structure. It consists of two *coupled* first-order loops, one from x_1 to u_1 and another x_2 to u_2 . A block diagram of this structure is shown in Figure 4.2. The x_1 to u_1 loop is sampled and updated every 0.8 seconds while the x_2 to u_2 loop is sampled and updated four time as often, every 0.2 seconds. These sampling rates are approximately 10 time the desired closed-loop bandwith for each loop. In addition, the compensator accounts for a one-half sampling period computational delay. It's sampling schedule is shown in Figure 4.3.

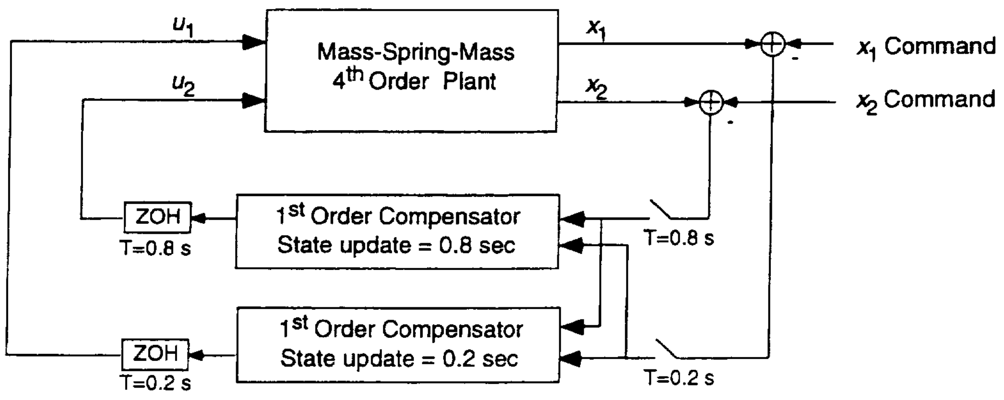


Figure 4.2 Block Diagram of Closed-Loop Multirate System

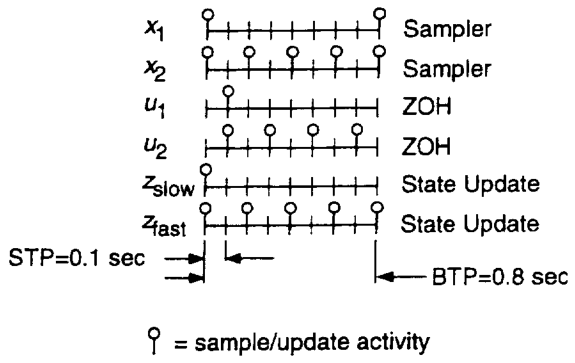


Figure 4.3 Sampling Schedule for Multirate Compensator

Step 3: Design a compensator with the sampling schedule and structure selected in Step 2 that stabilizes all plant perturbations. This will be the starting point for the optimization.

We designed a compensator using successive loop closures and root locus for the nominal plant. The compensator has the same structure and sampling schedule as the multirate design discussed in Step 2 except there is no input coupling and we did not account for computational delay. This compensator does however stabilize both the nominal and perturbed plants even with the computational delay and so can be used as the starting point for the optimization.

Refer to Reference 1 for an introduction to successive loop closures.

Step 4: Load the problem definition into Matlab's workspace variables defined in Table 3.2.

We defined the workspace variables using the Matlab script provided in Appendix D. To load the workspace variables for this example, type

```
»mropt_mkm
```

Step 5: Initialize the workspace variables and generate the necessary global variable using mropt_init

Type

»**`mropt_init`** *see Section 3.3.2*

At this point you can compute the minimum value of the cost function as follows (see Section 3.3.5)

»**`calc_LQGcost`** *see Section 3.3.5*

```
» ans =
    2.0108e+00
```
this is the minimum cost our design could achieve

Step 6: Calculate optimum values of the digital processor gains using mropt_optim

A partial output of the optimization follows

»**mropt_optim** *see Section 3.3.3*

```
Automatically selecting dscale
```
*a diagonal dscale is automatically selected if the variable **dscale** does not exist, or is not compatible with the current compensator*

```
Calculate gradient only? (y or n)n
```
*if we replied y the algorithm would compute only the gradient of the cost with respect to the free digital processor gains and the value of the cost for the current compensator gains in **cmp***

f-COUNT = *the total number of function evaluations*
FUNCTION = *the value of the cost function at the current iteration. Refer to Matlab FMINU documentation*

```
f-COUNT   FUNCTION    STEP-SIZE      GRAD/SD  LINE-SEARCH
    2         176         0.01    -9.48e+07
-> Plant  1 unstable while calculating J
-> Plant  1 unstable while calculating J
-> Plant  1 unstable while calculating J
-> Plant  1 unstable while calculating J
-> Plant  1 unstable while calculating J
-> Plant  1 unstable while calculating J
-> Plant  1 unstable while calculating J
-> Plant  1 unstable while calculating J
```
The algorithm encountered a destabilizing solution. It automatically adjusts the step size and continues the optimization

```
  -9.9655e-01
  -1.6399e-02
   1.0030e+00
  -1.0676e-02
  -2.2977e-02
  -9.4363e-01
  -1.3703e-03
   1.0529e+00
   1.0030e+00
   1.0036e+00
   1.0529e+00
   1.0085e+00
        .
        .
        .
```
these are the values of the scaled digital processor gains at the current iteration

```
   313          4.62          14.2      2.45e-13  int_st
 Optimization Terminated Successfully
 Gradient less than options(2)
  NO OF ITERATIONS=314
   -8.0984e-01
   -1.6700e+01
    1.8697e+00
    2.2849e+01
    7.4219e-01
   -7.3479e+00
   -8.6882e+00
    7.9308e+00
    1.0706e+00
    3.8424e-03
    1.0367e+00
    1.4037e+00

 ****     Final Results     ****
       Gain        Gradient
   -----------   -----------
   -1.6197e-01  -4.6527e-06
   -1.6700e+01   6.6547e-04
    1.8697e+00  -2.8675e-04
    2.2849e+01   4.5050e-04  these are the unscaled digital processor gain values
    7.4219e-01   1.6576e-04  and their scaled gradient values
   -5.5109e+00  -8.9955e-04
   -8.6882e+00  -4.8353e-05
    7.9308e+00  -8.5171e-04
    8.5648e-02  -1.0838e-04
    1.9212e-03   6.9640e-04
    6.2202e-01   3.4133e-03
    1.4037e-01   3.2564e-04

Final cost = 4.62482          the optimum value of the cost function

Optimized compensator gains

cmpf =

   1.9212e-03            0   1.8697e+00  -8.6882e+00
            0   1.4037e-01   2.2849e+01   7.9308e+00
   8.5648e-02            0  -1.6197e-01   7.4219e-01
            0   6.2202e-01  -1.6700e+01  -5.5109e+00

Elapsed time: 1516.05 sec
```

4.2 DESIGN ANALYSIS

We looked at two criteria when evaluating our multirate design: 1) the final cost; and 2) the step response to a command input.

4.2.1 Final Cost

The final cost for the multirate compensator is calculated automatically by the optimization routine **mropt_optim**. The LQG cost is computed by the function **calc_LQGcost** in step 5. For our system

$$J_{\text{multirate}} = 4.6 \quad \text{and} \quad J_{\text{LQR}} = 2.0$$

4.2.2 Step Response

We obtained the response to a unit command step to x_1 as follows

```
»[a,b,c,d]=mkm;  get the nominal plant
»b=[b zeros(4,1)];c=[c;[1 0 0 0]];  create a reference input
»d=[d zeros(4,1);0 0 -1];
»ur=[zeros(2,100);ones(1,100)];     generate a step input function for the input
                                     and compute the time domain response, see Section 3.2.1
»[y,t]=mrsim([a b;c d],4,cmpf,2,su,sy,sz,Stppbtp,1,[1 2],[5 3],ur,stp);
                         assuming cmpf, su, sy, sz, Stppbtp, and stp were previously defined and
                         that cmpf contains the optimized values of the digital processor gains
»plot(t,y([1 3],:))
```

We similarly obtained the step response for the LQR design. The results are shown in Figure 4.4.

As expected the performance of the multirate design does not match that of the LQ regulator. The LQ regulator was optimized for only the nominal plant. The multirate designs on the other hand represent a compromise, stabilizing both plants and providing "optimum" performance for both. In addition, the LQ design was a continuous full-state design whereas the multirate design was discrete second order and used only two state measurements.

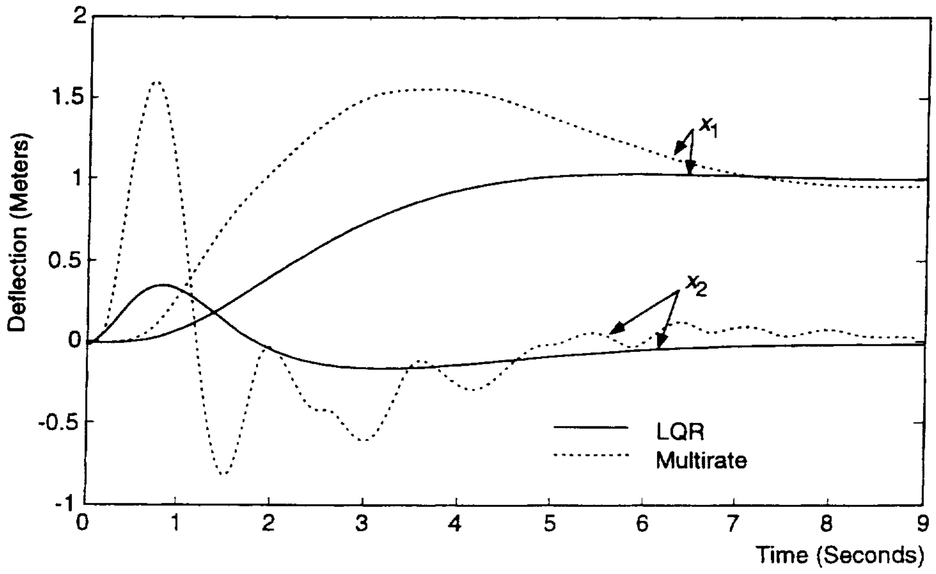


Figure 4.4 Response of MKM system to command step response to x_1

References

1. Mason, G. S., "Multirate Compensator Synthesis and Analysis," Ph.D. Dissertation, University of Washington, Seattle, Washington, 1992.

2. Mason, G. S., and Berg, M. C., "Reduced Order Multirate Controller Synthesis," *AIAA Journal of Guidance, Control and Dynamics,* Vol. 15, No. 3, May-June. 1992, pp. 700-706.

3. Mason, G. S., and Berg, M. C., "Multirate Flutter Suppression System Design for a Model Wing," *AIAA Journal of Guidance, Control and Dynamics,* Vol. 17, No. 6, Nov. -Dec. 1994, pp. 1267-1274.

4. Mason, G. S., Berg, M. C., and Yang, G. S., "A New Multirate Sample-Data Control Law Structure and Synthesis Algorithm," *AIAA Journal of Guidance, Control and Dynamics,* Vol. 15, No. 5, Sep.-Oct. 1992, pp. 1183-1191.

5. Mason, G. S., Berg, M. C., and Mukhopadhyay, V., "Multirate Flutter Suppression System Design for the Benchmark Active Controls Technology Wing: *Part I : Theory and Design Procedure,* NASA TM 2002-212128, December 2002.

Blank

APPENDIX A

GLOBAL WORKSPACE VARIABLES

The following global workspace variables are defined by **mropt_init**. Note that the first letter of every global variable is capitalized.

For each plant perturbation, the discrete closed-loop system can be written in the form:

$$x_i(m,n+1) = F_i x(m,n) + G_i u(m,n) + W w_i \tag{A.1}$$

$$y_i(m,n) = H_i x(m,n) + V w_i \tag{A.2}$$

$$u_i(m,n) = [S1(n)P(n)S2(n) + S3(\mathrm{n})]\, y_s(m,n) \tag{A.3}$$

where

$$F_i = \begin{bmatrix} \overline{F}_i & 0 \\ 0 & 0 \end{bmatrix},\ G_i = \begin{bmatrix} \overline{G}_i & 0 \\ 0 & I \end{bmatrix},\ W = \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix},\ H_i = \begin{bmatrix} \overline{H}_i & 0 \\ 0 & I \end{bmatrix},\ V = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix},\ P = \begin{bmatrix} D_z & C_z \\ B_z & A_z \end{bmatrix} \tag{A.4}$$

and w is a discrete zero mean white noise input with

$$E\{w_i(m,n)w_i^{\mathrm{T}}(m+j,n+k)\} = \{\delta(k)+\delta(j)\}R_i \tag{A.5}$$

The subscript i indicates the i^{th} plant condition and the overbarred matrices are the discretized plant matrices. The matrices $S1$, $S2$, and $S3$ are periodically time-varying switching matrices which model the multirate sampling scheme, and P is a matrix of the compensator's digital processor gains.

The discrete cost function for the system in Eqns. (A.1-A.3) can be written as

$$J = \lim_{m\to\infty} \frac{1}{N}\sum_{i=1}^{\mathrm{Np}}\sum_{n=0}^{N-1} E\left\{\begin{bmatrix} x_i(m,n) \\ u_i(m,n) \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} Q_{1i} & M_i \\ M_i^{\mathrm{T}} & Q_{2i} \end{bmatrix} \begin{bmatrix} x_i(m,n) \\ u_i(m,n) \end{bmatrix}\right\} \tag{A.6}$$

See Reference 1 for more information.

The global Matlab workspace variables are then defined as follows.

Plant Description:

Np = number of plant conditions

MF = $[F_1, F_2, ..., F_{\mathbf{Np}}]$	**NcF** = number of columns in F_i
MG= $[G_1, G_2, ..., G_{\mathbf{Np}}]$	**NcG**= number of columns in G_i
MW= $[W_1, W_2, ..., W_{\mathbf{Np}}]$	**NcW** = number of columns in W_i
MH= $[H_1, H_2, ..., H_{\mathbf{Np}}]$	**NcH**= number of columns in H_i
MV= $[V_1, V_2, ..., V_{\mathbf{Np}}]$	**NcV** = number of columns in V_i
MR= $[R_1, R_2, ..., R_{\mathbf{Np}}]$	**NcR** = number of columns in R_i

Compensator Description:

Nzhcmp = number of states in the periodically time-varying model of the multirate compensator

Stppbtp = the global version of **stppbtp**

S1k = [$S1(0)$, $S1(0)$, ..., $S1(N\text{-}1)$]	**Ns1** = the number of columns in $S1$
S2k = [$S2(0)$, $S2(0)$, ..., $S2(N\text{-}1)$]	**Ns2** = the number of columns in $S2$
S3k = [$S3(0)$, $S3(0)$, ..., $S3(N\text{-}1)$]	**Ns3** = the number of columns in $S3$

Ptv: abs(**Ptv**) = number of columns in the compensator gain matrix $[A_z, B_z; C_z, D_z]$. **Ptv** is positive if the digital processor gains are time-invariant and negative if they are periodically time-varying.

Cost Function Description:

MQ1= $[Q_{11}, Q_{12}, \dots Q_{1\mathbf{Np}}]$	**NcQ1**= number of columns in Q_{1i}
MQ2= $[Q_{21}, Q_{22}, \dots Q_{2\mathbf{Np}}]$	**NcQ2**= number of columns in Q_{2i}
MM= $[M_1, M_2, \dots M_{\mathbf{Np}}]$	**NcM**= number of columns in M_i

Miscellaneous Definitions:

Ppn , **Ppm**, and **Ppinit** : used internally

Ppfree = index to the "free" digital processor gains

Fcl = closed-loop state transitions matrix of the last computed ETIS system

Xc_global = value of the compensator gains at the last iteration

Unstable:	If **Unstable** =0	the compensator gains at the current iteration stabilize the plants
	Otherwise	they destabilize the plant

APPENDIX B

SUPPORT M-FILES

B.1 MODELING SUPPORT M-FILES

B.1.1 mrsplit

Format: **[a, b, c, d] = mrsplit(P, n)**

Description: Separates the composite gain matrix **P** into its constituents, where

$$\mathbf{P} = \begin{bmatrix} \mathbf{d} & \mathbf{c} \\ \mathbf{b} & \mathbf{a} \end{bmatrix} \tag{B.1}$$

Inputs: **P** see Eqn. (B.1)
n the number of columns in **a**

Outputs: **a, b, c, d** the gain matrices in Eqn. (B.1)

B.1.2 mrmakesk

Format: **[s1k, ns1k, s2k, ns2k, s3k, ns3k]=mrmakesk(su, sy, sz, stppbtp)**

Description: The periodically time-varying representation of a multirate compensator in Eqn. (2.2) can be written as

$$\begin{bmatrix} y(m,n) \\ x(m,n+1) \end{bmatrix} = \begin{bmatrix} D(n) & C(n) \\ B(n) & A(n) \end{bmatrix} \begin{bmatrix} u(m,n) \\ x(m,n) \end{bmatrix} = \left\{ S1(n) \begin{bmatrix} D_Z(n) & C_Z(n) \\ B_Z(n) & A_Z(n) \end{bmatrix} S2(n) + S3(n) \right\} \begin{bmatrix} u(m,n) \\ x(m,n) \end{bmatrix} \tag{B.2}$$

where A_Z, B_Z, C_Z and D_Z are the digital processor gains and $S1$, $S2$ and $S3$ are switching matrices which model the compensator's sampling schedule. **mrmakesk** creates the periodically time-varying switching matrices given the sampling schedule description.

Inputs: **su, sy, sz, stppbtp** see Table 3.1

Outputs: **s1k** = [$S1(0)$ $S1(1)$ ... $S1($**stppbtp**-1$)$]
s2k = [$S2(0)$ $S2(1)$ ... $S2($**stppbtp**-1$)$]
s3k = [$S3(0)$ $S3(1)$ ... $S3($**stppbtp**-1$)$]
ns1k, ns2k and **ns3k** the number of columns in $S1(\cdot)$, $S2(\cdot)$ and $S3(\cdot)$ respectively

B.1.3 mrgetsk

Format: **[s1, s2, s3]=mrgetsk(k, s1k, ns1k, s2k, ns2k, s3k, ns3k)**

Description: Extracts the individual switching matrices for a given STP from **s1k**, **s2k** and **s3k**.

Inputs: **k** an integer specifying for which STP the switching matrices are to be extracted
s1k, ns1k, s2k, ns2k, s3k, ns3k see Section B.1.2.

Outputs: **s1, s2, s3** switching matrices corresponding to $S1($**k**$)$, $S2($**k**$)$ and $S3($**k**$)$ respectively. See Section B.1.2.

B.2 SYNTHESIS SUPPORT M-FILES

B.2.1 calc_djdp

Format: **djdp=calc_djdp(pguess, dscale)**

Description: Used by the routine **mropt_optim** to calculate the derivative of the cost J with respect to the free compensator parameters. See Section 2.3.

Inputs: **pguess** a vector of the values of the free digital processor gains at the current iteration
dscale see Table 3.3.

Comments: This routine uses the global workspace variables generated by **mropt_init**

B.2.2 calc_j

Format: **djdp=calc_j (pguess, dscale)**

Description: Used by the routine **mropt_optim** to calculate the cost J at the current iteration

Inputs: **pguess** a vector of the values of the free digital processor gains at the current iteration
dscale see Table 3.3.

Comments: This routine uses the global workspace variables generated by **mropt_init**

B.2.3 calc_L

Format: **[lacc]= calc_L(pk, f, g, w, h, q1, q2, m)**

Description: Called by **calc_djdp** to calculate the steady-state values of a Lagrange multiplier. (The multiplier is used to adjoin the stability equality constraints to the cost function.)

Inputs: **pk** gains for the periodically time-varying representation of the multirate compensator

$$\mathbf{pk} = [P(0), P(1), \ldots, P(N-1)] \text{ and } P(i) = \begin{bmatrix} D(i) & C(i) \\ B(i) & A(i) \end{bmatrix} \tag{B.3}$$

$D(i)$, $C(i)$, $B(i)$, and $A(i)$ are the state space matrices for the periodically time varying representation of the compensator. See Section 2.2 and Section B.1.2

f, g, w, h plant description matrices for the current plant perturbation corresponding to F_i, G_i, W and H_i in Eqn. (A.4).

q1, q2, m cost weighting matrices for the current plant perturbation corresponding to Q_{1i}, Q_{2i} and M_i in Eqn. (A.6).

Output: **lacc** periodically time varying lagrange multipliers where

$$\mathbf{lacc} = [\Lambda(1)\ \Lambda(2) \ldots \Lambda(N\text{-}1)\ \ \Lambda(0)]$$

and $\Lambda(i)$ is the lagrange multiplier for the i^{th} STP

Comments: This routine uses the global workspace variables generated by **mropt_init**

B.2.4 calc_P

Format: **[pk]=calc_P(p)**

Description: Called by **calc_j** and **calc_djdp** to compute the periodically time-varying representation of the multirate compensator

Inputs: **p** the digital processor gains in the form

$$\mathbf{p} = [p(0), p(1), \ldots, p(N\text{-}1)] \text{ where } p(i) = \begin{bmatrix} D_z(i) & C_z(i) \\ B_z(i) & A_z(i) \end{bmatrix}$$

Outputs: **pk** matrix of the compensator gains for the periodically time-varying representation of the multirate compensator

$$\mathbf{pk} = [P(1), P(2), P(3), \ldots P(N)] \text{ where } P(i) = S1(i)p(i)S2(i) + S3(i)$$

See Eqn. (A.2) or Section B.1.2.

Comments: This routine uses the global workspace variables generated by **mropt_init**

B.2.5 calc_X

Format: **[xacc]= calc_X(pk, f, g, w, h, v, r)**

Description: Called by **calc_djdp** and **calc_j** to calculate the steady-state covariance values of the closed-loop system

Inputs: **pk** matrix of the compensator gains for the periodically time-varying representation of the multirate compensator, see Section B.2.4.

f, g, w, h plant description matrices for the current plant perturbation corresponding to F_i, G_i, W, H_i and V in Eqn. (A.4) respectively

r process and sampling noise covariances for the current plant perturbation corresponding to R_i in Eqn. (A.5).

Comments: This routine uses the global workspace variables generated by **mropt_init**

B.2.6 disc_cost

Format: **[q1d, q2d, md]=disc_cost(plt, nplt, stp, q1, q2, m)**

Description: Computes the discrete cost function weighting matrices such that the cost associated with a continuous plant in feedback with a single-rate compensator is identical to the cost associated with the discretized plant in feedback with the single-rate compensator. The continuous cost function is given in Eqn. (2.6); the discrete cost function is given by

$$J_{discrete} = \lim_{m \to \infty} \frac{1}{N} \sum_{n=0}^{N-1} E\left\{ \begin{bmatrix} x(m,n) \\ u(m,n) \end{bmatrix}^T \begin{bmatrix} Q_{1d} & M_d \\ M_d^T & Q_{2d} \end{bmatrix} \begin{bmatrix} x(m,n) \\ u(m,n) \end{bmatrix} \right\} \qquad \text{(B.4)}$$

disc_cost assumes the control input is updated using a zero-order-hold.

Inputs: **plt, nplt, stp** see Table 3.1

q1, q2, m the continuous cost function weighting matrices in Eqn. (2.6)

Outputs: **q1d, q2d, md** the discrete weighting matrices in Eqn. (B.4)

B.2.7 disc_noise

Format: [**w**]=**disc_noise**(**wxx**, **plt**, **nplt**, **stp**)

Description: Computes the covariance of a discrete-time white noise process disturbance such that the expected value of the states of a continuous plant, and the expected value of the states of the discretized plant are equivalent at the sampling instances. Thus, given the continuous plant

$$\dot{x}(t) = Ax(t) + G_w w(t), \text{ where } E(w(t)w^T(t+\tau)) = \delta(\tau)\mathbf{wxx}$$

and the plant discretized with a zero-order-hold at a sampling period T

$$x_d(m,n+1) = A_d x_d(m,n) + w_d(m,n), \text{ where } E(w_d(m,n)w_d^T(m+k,n+j)) = (\delta(k)+\delta(j))\mathbf{w}$$

then $E\{x_d(m,n)x_d^T(m,n)\} = E\{x((mN+n)T)x^T((mN+n)T)\}$

Inputs: **wxx** PSD of the continuous processor noise
plt, **nplt**, **stp** see Table 3.1. The inputs to **plt** are assumed to be only the process noise

Outputs: **w** PSD of the discrete process noise

B.2.8 dlyap2

Format: **x**=**dlyap2**(**a**, **c**, **method**)

Description: Solves the discrete lyapunov equation

$$\mathbf{x} = \mathbf{a}\cdot\mathbf{x}\cdot\mathbf{a}^T + \mathbf{c} \tag{B.5}$$

Inputs: **a**, **c** system matrices in Eqn. (B.5)
if **method** = 'Bartels' then the algorithm solves Eqn. (B.5) using Bartels method
otherwise Eqn. (B.5) is solved using partial sums

Outputs: **x** the solution to the lyapunov equation

B.2.9 get_cost

Format: [**q1**, **q2**, **m**]=**get_cost**(**i**)

Description: Extracts the cost function weighting matrices $Q1$, $Q2$ and M for the **i**th plant perturbation from the global workspace variables **MQ1**, **MQ2**, and **MM**. See Appendix A.

Inputs: **i** an index to the desired plant perturbation

Outputs: **q1**, **q2**, **m** the cost weighting matrices corresponding to $Q1$, $Q2$ and M respectively

Comments: This routine uses the global workspace variables generated by **mropt_init**

B.2.10 get_plt

Format: **[f, g, w, h, v, r]=get_plt(i)**

Description: Extracts the plant matrices F_i, G_i, W, H_i, V, and R_i for the **i**th plant perturbation from the global workspace variables **MF**, **MG**, **MW**, **MH**, **MV** and **MR**. See Appendix A.

Inputs: **i** an index to the desired plant perturbation

Outputs: **f**, **g**, **w**, **h**, **v**, **r** plant description matrices for the **i**th plant perturbation corresponding to F_i, G_i, W, H_i, V, and R_i respectively

Comments: This routine uses the global workspace variables generated by **mropt_init**

B.2.11 get_ppfree

Format: **ppfree = get_ppfree(cmpfree, ncmp)**

Description: Called by **mropt_optim** to determine which compensator gains will be optimized

Inputs: **cmpfree, ncmp** see Table 3.1.

Outputs: **ppfree** vector whose elements point to the free elements in **cmp**

B.2.12 get_sk

Format: **[s1, s2, s3]=get_sk(k)**

Description: Extracts the switching matrices $S1(k)$, $S2(\mathrm{k})$ and $S3(k)$ for the **k**th STP from the global workspace variables **S1k**, **S2k**, and **S3k**. See Appendix A.

Inputs: **k** indicates the **k**th STP of the current BTP. NOTE: **k** = 1, 2, ... , **stppbtp**

Outputs: **s1**, **s2**, **s3** switching matrices for the **k**th STP corresponding to $S1(\mathbf{k})$, $S2(\mathbf{k})$ and $S3(\mathbf{k})$ respectively

Comments: This routine uses the global workspace variables generated by **mropt_init**

B.2.13 make_noise

Format: **[r]=make_noise(rw, rv)**

Description: Creates a compound noise covariance matrix from **rw** and **rv**

Inputs: **rw**, **rv** the process and sensor noise covariance matrices respectively

Outputs: **r** compound noise matrix of the form

$$\mathbf{r} = \begin{bmatrix} \mathbf{rw} & \mathbf{0} \\ \mathbf{0} & \mathbf{rv} \end{bmatrix}$$

B.2.14 make_plt

Format: **[f, g, w, h, v, nzcmp]=make_plt(pltc, nplt, ncmp, stp)**

Description: Creates the compound plant matrices F_i, G_i, W and V for the current plant perturbation. See Appendix A.

Inputs: **pltc** has the same form as **plt** in Table 3.1 except it describes only a single plant perturbation
nplt, ncmp, stp See Table 3.1.

Outputs: **f, g, w, h, v** compound plant matrices F_i, G_i, W and V for the current plant perturbation respectively
nzcmp the number states in the periodically time-varying representation of the multirate compensator

B.2.15 mropt_check

Format: **mropt_check**

Description: **mropt_check** performs elementary error checking on the data in Table 3.2 and set the variable **err**=1 if it finds an error in the data. **mropt_check** is called by **mropt_init.**

Inputs: None

Outputs: None

B.2.16 mropt_global

Format: **mropt_global**

Description: Called by **mropt_init** to define the global workspace variables

Inputs: None

Outputs: None

B.2.17 mropt_fminu

Format: [x, OPTIONS] = **mropt_fminu(FUN, x, OPTIONS, GRADFUN, P1, P2, P3, P4, P5, ...
P6, P7, P8, P9, P10)**

Description: A modified version of **FMINU** from the Optimization_Toolbox. This version automatically reduces the search step size when a destabilizing solution is encountered.

Inputs: See **FMINU**

Outputs: See **FMINU**

Comments: Requires the Optimization_Toolbox

APPENDIX C

M-FILE MKM

The following M-File creates the state space matrices for the Mass-Spring-Mass system of Section 4.0

```
function [a,b,c,d]=mkm(m1,m2,k,b)
%[a,b,c,d]=mkm(m1,m2,k,b)

if nargin~=4  %nominal plant
  m1=1;
  m2=.1;
  k=1;
  b=0.01;
end;

t=(1/m1)+(1/m2);

a=[0  1  0     0
   0  0  k/m1 b/m1
   0  0  0     1
   0  0 -k*t  -b*t];

b=  [0      0
     1/m1   0
     0      0
    -1/m1   1/m2];

c=eye(4);
d=zeros(4,2);
```

Blank

APPENDIX D

SCRIPT MROPT_MKM

The following script defines the workspace variables in Table 3.2 for the example problem of Section 4.0

```
% Script: mropt_mkm.m
% Example script for creating input data for mropt
%
% Creates the following data required by mropt_init
%   plt,nplt,wxx,rv,q1a,q2a,ma
%   ucol,ncol,srow,crow
%   sz,su,sy,cmp,stp,Stppbtp,cmpfree

% G. Mason, U.W. 1992

% ----- Continuous plant description -----

Np=2;         % The number of plant conditions
nplt=4;       % The number of states in the plants
plt=[];       % Clear the variable which holds the plant data

% Plant #1
[a,b,c,d]=mkm(1,0.1,1,0.01);
plt = [a b; c d];
% Plant #2
[a,b,c,d]=mkm(1,0.2,1,0.01);
plt = [plt [a b;c d]];

% Pointer to rows and columns in the plant
ucol=[1 2];       % control input
ncol=2;           % process noise input
srow=[1 3];       % sensor output from the plant
crow=[1 2 3 4];   % criterion output from the plant

% Continuous process noise PSD
% wxx is a compound matrix like plt, it contains wxx for plant #1 and #2
wxx=[.1  .1];

% Discrete sensor noise PSD
% with same compound form as plt
rv=eye(2)*0.001;
rv=[rv rv];

% ----- Continuous Cost weighting matrices -----
% Output criterion weights
% Use 1e-8 instead of 0 so that qa1 & qa2 are positive definite
% The synthesis algorithm would accept semi-definite q1a & q2a
% but calc_LQGcost will not

q1a1=diag([5.5 1e-8 1e-8 2.2]);
q1a2=diag([6.5 1e-8 1e-8 4.8]);
q1a=[qa1/10.6 qa2/8.0];

% Control input weights
q2a=diag([1e4 1e1]);
```

```
q2a=[ra/10.6 ra/8.0];

% Cross weighting (ma)
% if there is no cross weighting it can be left undefined
clear ma

% ----- Compensator description -----
% Digital processor gains from successive loop closures design
cmp=[ 0.5  0      1    0
      0    0.1    0    1
      0.08 0     -0.2  0
      0    0.6    0   -0.75];

% Sampling schedule
su=[0 0;1 1;0 0;0 1;0 0;0 1;0 0;0 1];
            % Compensator output updating w/ delay
sy=[8 2];   % Sensor input sampling, multiplexed
sz=[8 2];         % Compensator state update
stp = .1;         % The shortest sampling period
Stppbtp = 8;      % The number of stp's in one BTP

% Free compensator gains
cmpfree=[1 0 1 1
         0 1 1 1
         1 0 1 1
         0 1 1 1];

% clear temporary variables
clear a b c d
```

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)* 12-2002	**2. REPORT TYPE** Technical Memorandum	**3. DATES COVERED** *(From - To)*
4. TITLE AND SUBTITLE Multirate Flutter Suppression System Design for the Benchmark Active Controls Technology Wing *Part II: Methodology Application Software Toolbox*		**5a. CONTRACT NUMBER**
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Mason, Gregory S.; Berg, Martin C.; and Mukhopadhyay, V.		**5d. PROJECT NUMBER**
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER 706-17-51-03
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) NASA Langley Research Center Hampton, VA 23681-2199		**8. PERFORMING ORGANIZATION REPORT NUMBER** L-18249
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration Washington, DC 20546-0001		**10. SPONSOR/MONITOR'S ACRONYM(S)** NASA
		11. SPONSOR/MONITOR'S REPORT NUMBER(S) NASA/TM-2002-212129

12. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified - Unlimited
Subject Category 63
Availability: NASA CASI (301) 621-0390 Distribution: Nonstandard

13. SUPPLEMENTARY NOTES
An electronic version can be found at http://techreports.larc.nasa.gov/ltrs/ or http://techreports.larc.nasa.gov/cgi-bin/NTRS

14. ABSTRACT
To study the effectiveness of various control system design methodologies, the NASA Langley Research Center initiated the Benchmark Active Controls Project. In this project, the various methodologies were applied to design a flutter suppression system for the Benchmark Active Controls Technology (BACT) Wing. This report describes the user's manual and software toolbox developed at the University of Washington to design a multirate flutter suppression control law for the BACT wing.

15. SUBJECT TERMS
Cybernetics; Digital Control Systems; Multirate sampling; Flutter suppression; Control law design and optimization; Metlab toolbox

16. SECURITY CLASSIFICATION OF: a. REPORT	b. ABSTRACT	c. THIS PAGE	**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** STI Help Desk (email: help@sti.nasa.gov) **19b. TELEPHONE NUMBER** *(Include area code)*
U	U	U	UU	46	(301) 621-0390

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18